Algorithms for Massive Data Project

Andrea Beretta
ID: 27201A

Academic Year 2023-2024

Abstract

The goal of the project is to develop a system that identifies frequent skill itemsets within the LinkedIn Jobs & Skills dataset. This detector, which conducts a Market-Basket analysis, treats the sets of skills required for each job posting as baskets and employs the A-Priori algorithm to discover all frequent itemsets of size k. The resulting skills can subsequently be utilized for further analysis.

Contents

1 Introduction

The objective of this project is to identify all frequent sets of required skills within a dataset of job postings. To achieve this, we utilize the market-basket model.

The market-basket model represents a many-to-many relationship between two types of elements, referred to as "items" and "baskets." Originally developed for analyzing actual market baskets to identify which products were frequently purchased together, this model extends beyond retail contexts and is applicable to various domains, including textual data analysis.

To handle the data in a scalable manner, I implemented the A-Priori algorithm and built a detector for frequent skills using PySpark, the Python API for Spark.

This report is structured as follows: Section 2 addresses the composition and preprocessing of the dataset; Section 3 covers the stages of algorithm implementation; Section 4 presents and analyzes the results; Section 5 examines the scalability of the detector.

2 The Data

The data used in this analysis is part of the LinkedIn Jobs & Skills (2024) dataset, released under the ODC-By license, Version 1.0, and available on Kaggle. Specifically, the "job skills" file is employed. This dataset consists of nearly 1.3 million rows, with two columns: one containing the link to the job posting and the other listing the required skills for that position. In this context, each row represents a basket associated with a set of items, i.e., the required skills.

2.1 Preprocessing

To prepare the data for use in the market-basket model and for algorithm iteration, a preprocessing step is necessary. This involves checking for and removing null values, converting text to lowercase, and detecting each skill—whether composed of one or more words—by splitting the list of items at each comma. Additionally, any punctuation or unnecessary characters are removed from the dataset.

Before implementing the algorithm, i chose to represent each item as an integer. Since frequent-itemset algorithms need to maintain numerous counts as they process the data, there may not be sufficient memory to store all of these counts, potentially causing the algorithm to slow down significantly. Therefore, these algorithms have a limit on how many items they can handle. When dealing with textual items, it is more space-efficient to represent them as consecutive integers from 1 to n, where n is the number of distinct items. To accomplish this, I built a hash table that maps distinct skills to integers using a Python dictionary, where each skill is a key, and its corresponding value is an integer. This allowed me to convert items to numbers and vice versa, for example, when visualizing the results.

2.2 Insights

As previously mentioned, the skills in the dataset can consist of one or more words. The number of skills listed in each job announcement varies, and some key metrics are summarized in the following table.

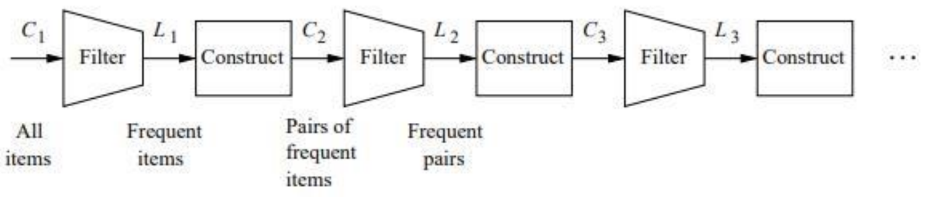
N. of Skills for Job	
Minimum:	1
Maximum:	463
Average:	21

3 The A-Priori Algorithm

The A-Priori algorithm is commonly chosen for frequent itemset mining because it reduces the number of candidates that need to be counted, thus lowering memory usage, albeit at the cost of performing additional passes over the data.

The A-Priori algorithm operates through "passes" to identify frequent itemsets, with one pass for each itemset size, k. I initially implemented each step of the algorithm individually and later developed an automatic function that replicates these steps. The process begins by analyzing the occurrences of individual items, given a support threshold, to identify which items are frequent as singletons. Based on the monotonicity property of itemsets—if a set I of items is frequent, then every subset of I must also be frequent—a pair cannot be frequent unless both its members are frequent. After the first pass, the function generates all distinct pairs consisting of two frequent items, which serve as candidates for the next iteration. Their occurrences are then evaluated to determine which pairs are frequent.

This process continues iteratively for each k-sized itemset, looping until no frequent itemsets of a certain size are found. At that point, monotonicity ensures that no larger frequent itemsets can exist, and the algorithm terminates. The progression from candidate set C to frequent itemset set L, moving from size k to size k + 1, is illustrated in the next picture



In the specific implementation of the function I developed, the user can freely select a percentage to compute the support threshold, as well as define a maximum itemset size to consider. For instance, setting this parameter to 2 means that only frequent singletons and pairs will be calculated (if applicable). For the purposes of this report, as mentioned earlier, the support threshold was set to 2%, and a small sample of the entire dataset was used. With these parameters, the algorithm takes approximately 8 minutes to execute.

It's important to note that, by definition, choosing a lower threshold would increase the algorithm's runtime and result in a larger number of frequent skills, which could potentially make the results less meaningful.

4 Results

The algorithm outputs a list of frequent itemsets. As a reminder, the support threshold was set to 2% of the baskets, meaning that for an itemset to be considered frequent, it must appear in at least 0.02 × the total number of job postings (baskets). In the sample used, the most frequent itemset is a singleton consisting of the skill "communication." The top six positions in the ranking are all occupied by singletons, which is expected due to the monotonicity property—frequent singletons are more likely to appear than larger itemsets.

In total, the A-Priori algorithm identified 83 frequent singletons, 69 pairs, and 22 triples. No 4-itemset was found, which triggered the algorithm to stop.

Let's take a closer look at some specific details:

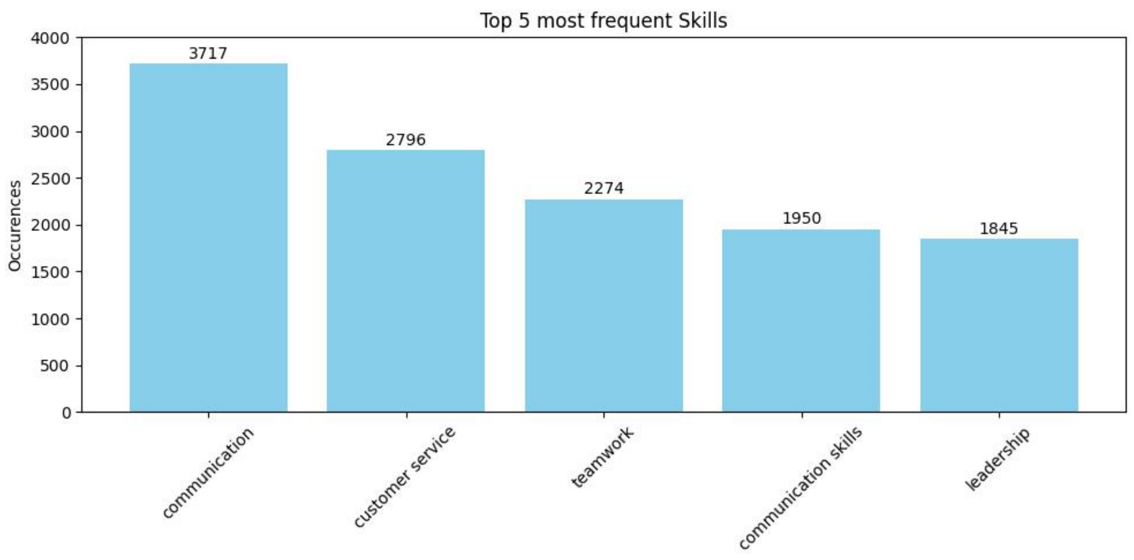


Figure 2: Most frequent singletons

The most frequent singletons include communication, customer service, teamwork, communication skills, and leadership. These are broad and general skills, offering little specific insight. However, the high frequency of "customer service" may indicate a significant number of jobs requiring direct interaction with clients, which suggests a demand for roles involving customer-facing responsibilities.

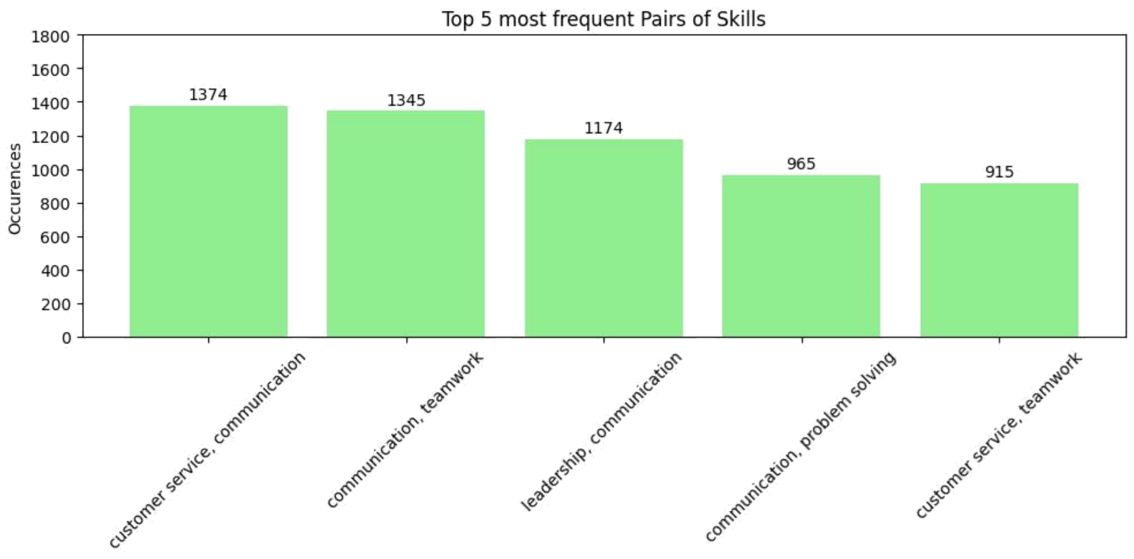


Figure 3: Most frequent pairs

Examining the frequent pairs reveals more interesting patterns. Four out of the five most frequent pairs include the skill "communication." This indicates that the most common pairs are largely combinations of the most frequent singletons, which is a direct consequence of the A-Priori algorithm's functioning.

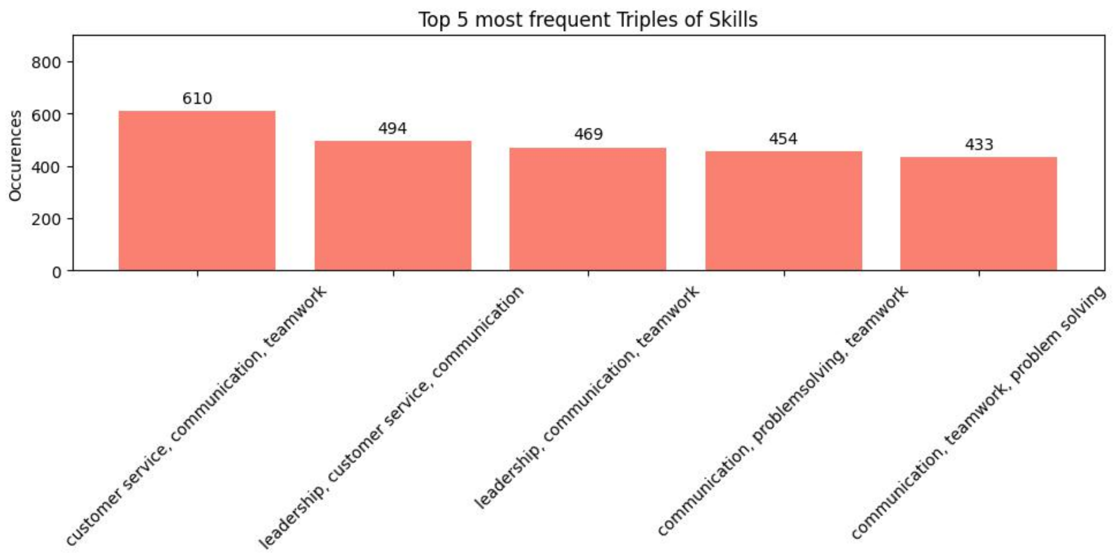


Figure 4: Most frequent triples

A similar pattern is observed in the frequent triples. The most common 3-itemset is the combination of "customer service," "communication," and "teamwork."

The frequent itemsets identified by the A-Priori algorithm can be utilized as input for further analysis with other portion of the data.

5 Scalability

As mentioned earlier, only a sample of the entire dataset was used for the experiment. However, the code was designed with scalability in mind, operating within a Spark-Hadoop environment using the PySpark API for Python.

The data is loaded into a Resilient Distributed Dataset (RDD), and the algorithm employs PySpark RDD functions such as Map, Reduce, and GroupByKey to ensure parallel processing and scalability. The algorithm scales effectively with data size, though the runtime may increase significantly depending on the available computational resources.

6 Conclusion

In this experimental project, I developed a detector for frequent skills in job announcements using the A-Priori algorithm. The detector was implemented in Python within the PySpark environment, with a focus on ensuring scalability.

The primary challenges involved working with Spark syntax and avoiding operations that could lead to computational overloads. While the algorithm is efficient, it remains computationally intensive and its runtime varies depending on the chosen support threshold and the volume of data processed.

7 References

1. Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman, ”Mining of Massive Datasets”, 2011